UCSD Jacobs | Computer Science and Engineering

Reducing Peak Power with a Table-Driven Adaptive Processor Core

Vasileios Kontorinis † , Rakesh Kumar ‡ , Dean Tullsen †

†University of California, San Diego, ‡University of Illinois Urbana-Campaign

1. Introduction

Average power(total energy over time) drives:

1. cooling costs
2. reliability

Receives much attention from research community. Thoroughly explored subject

Peak power(accumulation of power for each component with peak activity factors) impacts:

1. processor design
2. packaging
3. power delivery

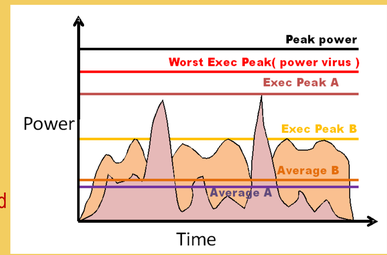
Highly under-studied problem. Still very important

Our approach:

➢ Employ a highly-adaptive core which consists of components configured at different levels.

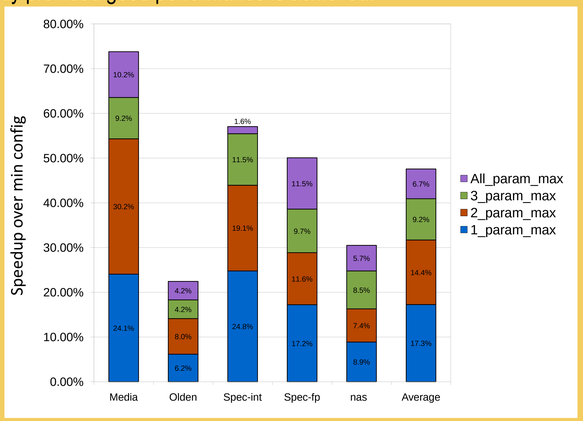
➢ The architecture guarantees not all of them are maximally configured at the same time.

➢. ***Peak power*** can be reduced with limited performance loss.



2. Motivation – The power of Adaptation

Most applications have few resource bottlenecks. As long as those resources are sufficiently provided good performance is achieved.



Maximizing 3 out of 10 resources will give more than 85% of the performance when all resources are max.

- Different benchmarks have different bottlenecks
- Hard to design the ideal processor for a wide range of applications.

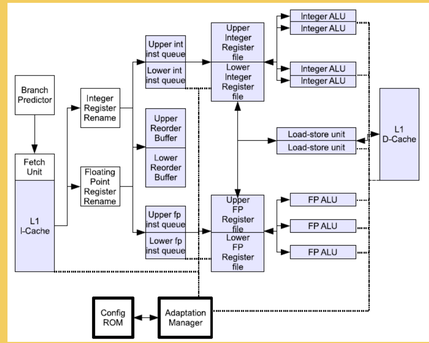
With efficient online resource adaptation:

➔ *Performance* close to maximally configured core

➔ *Peak power* close to minimally configured resources.

3. Architecture

- Predetermined processor configurations exist in *Config ROM*
- *Adaptation Manager* chooses configuration and power-gates/power-enables parts of processor
- Careful transitions provide peak power guarantees



INT instruction queue	16,32 entries
FP instruction queue	16,32 entries
INT registers	64,128
FP registers	64,128
INT alus	2,4
FP alus	1,2,3
Load/Store Units	1,2
Reorder Buffer	128,256 entries
Instruction Cache	4,8,16,64 KB
Data Cache	4,8,16,64 KB

The design space.
6144 configurations in total. Can be reduced to ~250 architecturally reasonable configurations per peak power threshold.
We maintain one ROM per peak power threshold.

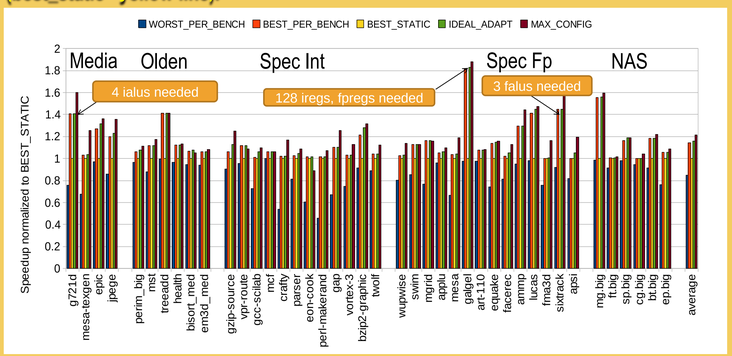
4. Other Potential Applications

1. Can be used in conjunction with Dynamic Thermal Management
2. Thermal hot spot avoidance
3. We can counteract the effects of process variation. During testing/verification, we fill the ROM accordingly
4. We can maintain overall peak power envelope and use mechanism to maximize throughput. Allow a core, that translates additional power to additional performance, to have peak power of P + Δ, while restricting another core, that does not benefit from additional power to P – Δ. (multiple ROMS / programmable ROM needed)

5. Static Policy Results

General purpose processors are optimized for the common case.

With adaptation at 70% of total peak power, individual benchmarks can perform up to 80% better *(ideal_adapt-green line)* than the best static configuration (best_static –yellow line).



Best Static configuration
iqs:16 fqs:32 ialu:2 falu:1 ldst:1 ics:32 dcs:16 ipr:64 fpr:64 rob:128

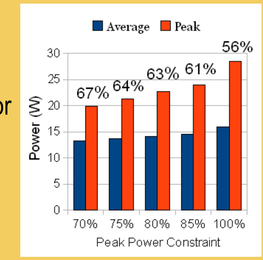
6. Power Efficiency

The more we restrict peak power the more we bridge the gap between average and peak power.

Low ratio ➔ Over-provisioned processor

High ratio ➔ More efficient design

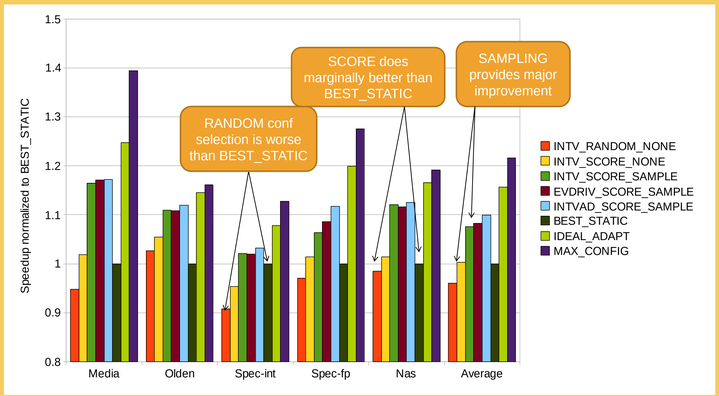
Both peak and average power become lower when we cap the peak power budget.



7. Realistic Dynamic Policy Results

Dynamic policies answer the following questions:

When to adapt ?	Which configuration to choose ?	How to evaluate a configuration ?
INTV: every fixed interval of cycles (2M cycles)	*RANDOM:* randomly pick the next configuration	*NONE:* do not evaluate configurations.
EVDRIV: when the IPC or the cache misses change by more than 30%	*SCORE:* evaluate each configuration according to which provides more of the bottleneck resource. Choose the highest score	*SAMPLE:* sample several configurations and peak the one with highest instructions per cycle. (when combined with SCORE, you sample the ones with highest score)
INTVAD: using an adaptive interval which shrinks when we find good configurations and extends when we do not. (0.5M – 8M cycles)		



INTVAD technique gives best results across all the benchmarks. At 70% of peak power covers 10% out of the 16% of the ideal oracle adaptation policy.

We can save 25% of peak power, for only 4% performance loss.